VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2012 Jiří Otáhal

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jiří Otáhal**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: Hella Corporate Center Central & Eastern Europe
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Krömer, Ph.D.**

Konzultant bakalářské práce: Ing. Lubomír Beneš

Datum zadání: 18.11.2011

Datum odevzdání: 04.05.2012

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Datum: 27. 4. 2012 Podpis:

Poděkování

Rád bych poděkoval panu Ing. Pavlu Krömerovi, Ph.D. za vedení během vypracovávání mé bakalářské praxe. Dále bych rád poděkoval osazenstvu kanceláře, kde jsem vykonával praxi, za příjemné pracovní prostředí a ochotnou spolupráci. Zvláštní dík patří pánům Jaroslavu Unzeitigovi, Ing. Lubomíru Benešovi a Ing. Vítězslavu Tománkovi za odborné vedení během vykonáváni praxe.

Abstrakt

Účelem této práce je popsat průběh individuální odborné praxe, kterou jsem vykonával během dvou semestrů třetího ročníku bakalářského studia. Práce obsahuje popis úkolu, který mi byl svěřen a postup jeho řešení. Dále je v ní obsažen výčet znalostí a dovedností, jež jsem potřeboval pro splnění daného úkolu. Na závěr je zmíněn přínos, který mi dala tato odborná praxe.

Mým hlavním úkolem při vykonávání odborné praxe bylo vytvořit program pro dálkový odečet hodnot momentální hmotnosti ze sil na granulát umělé hmoty pro vstřikovací lisy. Program byl řešen jako modul, který rozšiřoval funkčnost stávajícího programu o komunikaci s novými sily.

Klíčová slova

ModBus, Profibus, Java, Ethernet, TCP/IP, VNC, Framework

Abstract

The aim of this thesis is to describe the course of individual professional practice in the company performed throughout the two semesters of the third year of my bachelor's studies. The thesis contains a description of the tasken trusted to me and the approach to its resolution. It further contains a list of the knowledge and skills needed to fulfill the given task. The conclusion mentions the benefit given to me by this work.

My main task in performing this specialized research was to create a program for the remote reading of the values of current mass forces on granules of plastic for an injection molding machine. The program was created as a model which expanded the functionality of the pre-existing program to communicate with new forces.

KeyWords

ModBus, Profibus, Java, Ethernet, TCP/IP, VNC, Framework

Seznam použitých zkratek

ADU	Application Data Unit	Základní rámec protokolu ModBus
DHCP	Dynamic Host Configuration Protocol	Protokol pro automatické přidělování IP adresy, masky podsítě, výchozí brány a DNS serveru
ERP	Enterprise Resource Planning	Podnikový informační systém
ISO/OSI	International Organization for Standardization/Open Systems Interconnection	Referenční model sítí
JNLP	Java Network Launching Protocol	Spouštěcí soubor JWS, definující pomocí XML jak spustit JWS aplikaci
JWS	Java Web Start	Technologie distribuce Java aplikací po internetu
PDU	Protocol Data Unit	Část rámce protokolu ModBus
Profibus	Process Field Bus	Průmyslová sběrnice
SCS	Silo Control System	Řídicí systém sila
TCP	Transmission Control Protocol	Protokol transportní vrstvy
UDP	User Datagram Protocol	Protokol transportní vrstvy
VNC	Virtual Network Computing	Technologie pro dálkové sdílení pracovní plochy pomocí počítačových sítí
XML	Extensible Markup Language	Obecný značkovací jazyk

Obsah

1 Úvod

Účelem této práce je popsat individuální odbornou praxi, kterou jsem absolvoval ve firmě Hella Corporate Center Central & Eastern Europe a zhodnotit jaký přínos pro mne tato praxe měla. Práce popisuje úkol, který mi byl zadán pro splnění, jeho časovou náročnost a zvolený postup, který jsem zvolil pro jeho splnění. Na závěr je přidáno celkové zhodnocení dosažených výsledků. Součástí práce je také popis znalostí získaných během studia, které jsem uplatnil při řešení zadaného úkolu. Jsou zde popsané i znalosti, jenž mi pro splnění úkolu chyběly a které jsem musel získat.

Druhá kapitola pojednává o popisu firmy, kde jsem odbornou praxi vykonával. Její odborné zaměření a výčet služeb, které nabízí svým partnerům. Součásti kapitoly je i popis mého pracovního zařazení ve firmě během vykonávaní praxe. Třetí kapitola pojednává o zadaném úkolu, navíc je zde zadaný úkol rozdělen na menší dílčí části, které jsou stručně popsané. Kapitola je doplněna i časovou náročnost jednotlivých dílčích úkolů. Čtvrtá kapitola pojednává o samotném řešení zadaného úkolu. Celá kapitola je ještě rozdělena na popis postupu při řešení jednotlivých dílčích úkolů. Svým obsahem se tedy jedná o nejrozsáhlejší kapitolu. Pátá kapitola pojednává o znalostech nabytých během studia, nutných pro splnění zadaného úkolu. Šestá kapitola pojednává o znalostech, které bylo nutné si doplnit, abych mohl splnit zadaný úkol. Sedmá kapitola pojednává o celkovém zhodnocení dosažených výsledků a o tom jaký přínos pro mě tato praxe měla. Osmá, poslední, kapitola je věnována závěru, kde hodnotím dosažené výsledky a provádím celkový souhrn.

2 Popis společnosti

Tato kapitola má za úkol popsat činnost a zaměření firmy, u které jsem absolvoval odbornou praxí. Součástí kapitoly je i popis mého zařazení ve firmě během vykonávaní praxe.

2.1 Popis odborného zaměření společnosti

Společnost Hella Corporate Center Central &Eastern Europe je součástí koncernu Hella a zabývá se IT podporou, službami nákupu, finančními službami a dalšími službami pro region východní Evropy. Její IT oddělení zajišťuje podporu pro 15 společností sídlících v České republice, Slovensku, Polsku, Rakousku, Maďarsku, Slovinsku a Rumunsku. Kromě těchto zemí se podílí na celokoncernové podpoře, která zahrnuje i podporu v zavádění nových systémů a správu jak nově zavedených systémů, tak těch stávajících v koncernových společnostech v Asii i Americe. K této celokoncernové podpoře patří například podpora 7 dní 24 hodin, na které se podílí centrálami v Americe a Asii a střídají se podle chodu slunce. Služby poskytované těmto společnostem zahrnují místní podporu, D&D podporu, služby v oblasti datových center a sítí, konzultace ohledně SAP/ERP systému, služby Service Desk a vývoj Software pro potřeby výroby.

2.2 Popis pracovního zařazení studenta

Ve společnosti jsem byl zařazen jako programátor. Mým úkolem bylo navrhnout vhodné řešení programu pro dálkový odečet hodnot a jejich zobrazování z měřící váhy sila na plastový granulát a realizovat ho. Sice skoro polovina času stráveného na řešení zadané úlohy připadla na studium dokumentace jednotlivých zařízení, následném zkoumaní možností konektivity jednotlivých zařízení a použitých protokolů. Přesto jsem byl umístěn mezi programátory, protože i tato práce k tvorbě programu tohoto typu patří. Mým vedoucím ve společnosti byl pan Jaroslav Unzeitig, který mi zadával úkoly a poskytoval potřebné věci pro jejich řešení. Dále semnou na úkolu spolupracovali pánové Ing. Lubomír Beneš a Ing. Vítězslav Tománek, s nimiž jsem konzultoval jednotlivá technická řešení a technické problémy.

3 Seznam úkolu

Časová náročnost řešení úkolu	
Seznámení se s frameworkem	5
Studium dokumentace a protokolu	20
Implementace a testováni	25

Tabulka 2.1 Časová náročnost řešení úkolu

3.1 Zadaný úkol

Byl mi zadán úkol vytvořit program pro dálkový odečet hodnot momentální hmotnosti ze sil na granulát uměle hmoty pro vstřikovací lisy pro potřeby jak obchodního oddělení, tak pro potřeby výroby. Dosavadní situace byla taková, že hmotnost granulátu v silech umístěných venku mimo výrobní halu se dala odečítat jedině ze vzdáleného displeje, nainstalovaného v jedné provozní místnosti ve výrobě, nebo přímo ze snímače váhy a SCS v silech. Toto řešení neumožňovalo rychlou kontrolu množství granulátu v sile pro potřeby obchodního oddělení. Měl jsem tedy najít vhodné řešení, jak připojit jedno ze zařízení sila do podnikové sítě ethernet a zprovoznit přes toto připojení komunikaci pro možnosti dálkového odečtu hmotnosti. Řešení tohoto úkolu bylo rozděleno na dílčí úkoly:

3.1.1 Seznámení se s frameworkem

Program měl být řešen jako zásuvný modul pro firemní framework. Musel jsem se tedy seznámit se samotným řešením těchto modulů, jejich tvorbou a některými třídami z frameworku, které budu pro tvorbu samotného modulu využívat.

3.1.2 Studium dokumentace a protokolu

Abych mohl zjistit možnosti připojení jednotlivých zařízení umožňujících odečítání hmotnosti z vah sila, musel jsem prostudovat k nim dodanou dokumentaci. Jednalo se o zařízení snímač váhy, SCS a vzdáleného displeje. Z dokumentace jsem měl hlavně vyčíst portové vybavení jednotlivých zařízení a protokoly, které na těchto portech jsou implementovány a dále možnost těchto portu připojit do sítě ethernet, například pomocí vhodného převodníku.

3.1.3 Implementace a testování

Po vybrání vhodného zařízení, jsem měl za úkol implementovat protokol pro vybraný port v jazyce Java. Nejdříve jsem měl vytvořit testovací program pro ověření komunikace a správné funkčnosti vybraného protokolu. Následně po úspěšném otestovaní zkušebního programu, jsem začal s implementací zásuvného modulu s otestovaným komunikačním protokolem. Po dokončení zásuvného modulu jsem provedl další testování funkčnosti.

4 Zvolený postup řešení

4.1 Seznámení se s frameworkem

Jelikož program měl být řešen formou zásuvného modulu pro firemní aplikační rámec (framework), musel jsem se s ním nejdříve seznámit. Celý framework tvoří soubor různých tříd, které lze využit při tvorbě programu. Při spuštění programu postaveném na tomto frameworku se jako první spouští třída *BasicApplicationFrame*, která představuje základní frame. Tento frame obsahuje menu se základními položkami pro konfiguraci programu. Toto menu lze dále rozšířit o další položky. Dále se do tohoto frame nahrávají panely zásuvných modulů. Třída *BasicApplicationFrame* má dále dvě rozšíření, která umožňují spuštění programu přímo z webové stránky. První řešení využívá JWS. Tato technologie se liší od Javy appletu tím, že program neběží v prohlížeči, ale pomocí Web Start Tools jsou ze serveru staženy potřebné soubory a aplikace je následně spuštěna. Zároveň při každém spouštění se ověřuje dostupnost novější verze, to usnadňuje aktualizace programu. Samotný proces spouštění probíhá tak, že po kliknutí na odkaz pro spuštění aplikace odešle web server JNLP soubor, jenž zajistí spuštění JWS. V samotném souboru jsou uložená, ve formátu XML, data pro spuštění aplikace. JWS následně podle údajů obsažených v JNLP stáhne z datového serveru potřebné soubory a provede spuštění aplikace. Druhá varianta nazvaná HA Application využívá také JWS, ale dále ho rozšiřuje o možnost přesměrování na jiný web server v případě výpadku toho prvního. Pro tuto funkci jsou přidány parametry do JNLP souboru, obsahující seznam všech náhradních serverů. Proto nastane-li výpadek původního serveru, je komunikace plynule převedena na první funkční náhradní server ze seznamu.[1][2]

Framework taky obsahuje několik knihoven druhých stran. Jednou je knihovna FlexDock, která slouží pro manipulaci s panely modulu osazenými v hlavním frame. FlexDock umožňuje uživateli různě měnit velikost panelu, umístění panelu případně je zavírat, nebo připoutat k určitému místu. Rozmístění panelu se ukládá do konfiguračního souboru, pro znovunačtení rozmístění po opětovném startu. Další knihovnou je knihovna Apache log4j sloužící pro záznam log informací. V konfiguračním souboru lze pro tuto knihovnu nastavit druh logování. Apache log4j nabízí několik druhu logování, například logování do souboru nebo na příkazovém řádku. Pro tuto knihovnu existuje ve frameworku modul, který slouží k zobrazení logu přímo v aplikaci a tím usnadňuje ladění programu za běhu.

Samotné moduly se vytváří tak, že pro vytvoření třídy typu modul je nutné, aby tato třída dědila z abstraktní třídy *AbstractModule*. Tato třída obsahuje jednu abstraktní metodu initModule(), kterou musíme v novém modulu implementovat. Modul může obsahovat panely, akce a zařízení. Pro přidání panelu a akce slouží metody addPanel(JPanel panel, String id, String title) a addAction(Actionaction, String id), které obsahuje třída *AbstractModule*. Další třída, využitá při tvorbě programu, byla abstraktní třída *AbstractDevice*. Tato třída obsahuje dvě abstraktní metody initDevice() a closeDevice() a další metody pro práci se zařízením, jako je nastavení stavu nebo znovu připojení. Poslední třídou, kterou jsem používal při tvorbě programu, byla třída *PreferencesPanel*. Ta umožňuje uložení konfigurace do konfiguračního souboru pro znovunačtení konfigurace po restartu aplikace. Pro tyto účely obsahuje metody pro práci s preferences panelem, například načtení hodnot z komponent případně resetováni těchto hodnot. Třída, která dědí z této třídy, je považovaná za preference panel a je zařazena v podobě

záložky do okna preference. Pro vytváření preferences panelu se využívá vlastních komponent z frameworku.

4.2 Studium dokumentace a protokolu

4.2.1 Studium dokumentace

Než jsem mohl začít se samotnou tvorbou programu, musel jsem nejdříve prostudovat dokumentaci k snímači váhy, SCS a vzdálenému displeji. Tato zařízení jsou zapojená tak, že všechny váhy v silu jsou zapojeny do snímače váhy. Toto zařízení není vybaveno žádnými ovládacími prvky a jeho nastavení lze provést jedině pomoci RJ-45 konektoru, kde se lze připojit pomocí technologie VNC k vzdálenému grafickému uživatelskému rozhraní[4]. Pomocí něho jde přístroje kalibrovat a nastavit. Z tohoto důvodu je k snímači váhy připojeno zařízení SCS, které obsahuje ovládací prvky a umožňuje tak nastavení a kalibraci snímače váhy a dále také zobrazuje na svém displeji údaj o momentální hmotnosti granulátu v sile. Každé silo je vybaveno jedním snímačem váhy a jedním SCS. Zařízení SCS z jednotlivých sil jsou propojena pomocí sběrnice RMIC-NETBUS, pomocí které je k SCS připojen i vzdálený displej a umožňuje tak zobrazení momentálního stavu hmotnosti granulátu u jednotlivých sil.

Jako první jsem prostudoval dokumentací k SCS a vzdálenému displeji. Z dokumentace jsem zjistil, že obě zařízení jsou vybaveny konektorem RS485, do kterého je vyvedena průmyslová sběrnice Profibus a dále obě zařízení mohla být vybavena volitelně konektorem RJ-45 pro ethernet. Bohužel v dokumentaci nebyl konektor RJ-45 více popsaný, takže jsem nezjistil, jaký komunikační protokol podporuje. Následně po prozkoumání samotného zařízení jsem zjistil, že tyto volitelné konektory zařízení neobsahuje, tudíž jsem se možností konektoru RJ-45 dále nezabýval.[3]

Dále jsem prostudoval dokumentaci k snímači váhy. Toto zařízení je po stránce konektorového vybavení mnohem bohatší. Obsahuje analogové I/O, dále sériovou linku standartu RS232 a RS485 a nakonec i RJ-45, o kterém jsem se už zmiňoval ohledně VNC. Pro účely programu byly pro mne nejzajímavější konektory RS232 a RJ-45. Z dokumentace jsem zjistil, že konektor RS232 stejně jako RS485 podporuje několik komunikačních protokolu, které lze v nastavení pro daný konektor aktivovat. Jednalo se o protokoly ModBus, SMA a Asycom a dále bylo na těchto konektorech možné aktivovat tiskový port. Konektor RJ-45 kromě VNC a webového rozhraní podporoval komunikační protokol ModBus-TCP.[4]

Po prostudování dokumentací a možností konektorů jsem se dále zabýval studiem jednotlivých protokolů a sběrnic, abych zjistil možnosti jejich využití pro můj program. Jednalo se o sběrnici ProfiBus a o komunikační protokol ModBus využívající konektory RS232, RS485 a RJ-45.

4.2.2 Profibus

Profibus[5,6,7] je komunikační průmyslová sběrnice, která patří v dnešní době v automatizaci k nejčastěji používaným sběrnicím na světě. Její počátky sahají až do 80. let, kdy se několik významných německých firem dohodlo na společném vývoji nové průmyslové sběrnice. Tato sběrnice dostala název Profibus. Sběrnice vychází z otevřeného komunikačního modelu ISO/OSI a je určená pro všechny oblasti automatizace. Norma Profibusu není závislá na výrobci a její otevřenost je garantovaná normou IEC 61158 a IEC 61784.

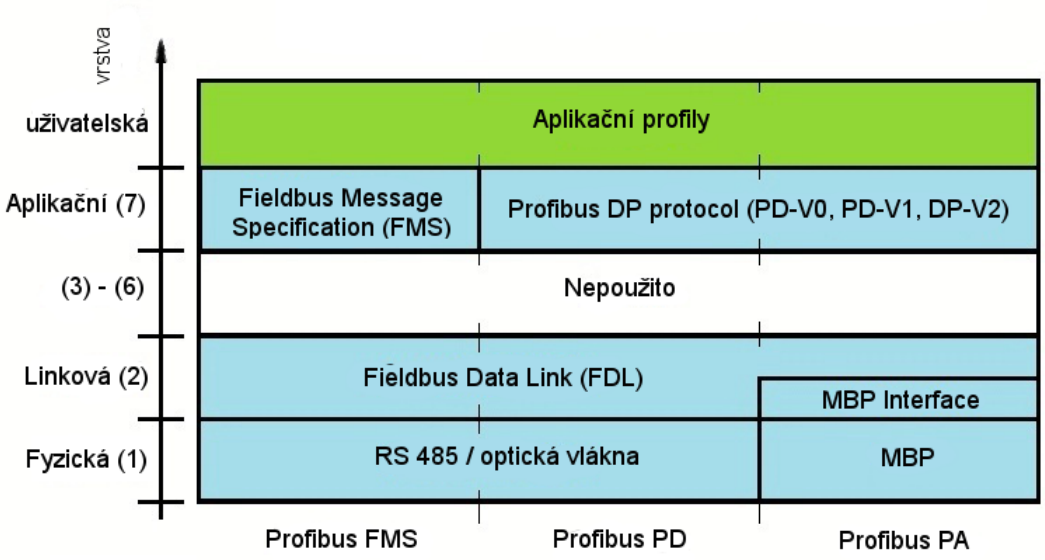
V dnešní době existují tři základní varianty komunikačního protokolu Profibusu:

- Profibus FMS, jedná se o první variantu Profibusu, která je určená pro náročné komunikační úlohy, pro které nabízí velkou skupinu služeb pro práci s daty, alarmy a programy. Oproti Profibusu PD dosahuje poněkud nižší maximální rychlosti, proto je díky větším komunikačním možnostem vhodnější pro nasazení na vyšších systémových úrovních, kde mezi sebou komunikují řídící systémy a systémy vizualizace a sběru dat.

- Profibus PD nejjednodušší a nejrozšířenější varianta Profibusu. Používá se pro komunikaci mezi centralizovanými zařízeními automatizace a decentralizovanými přístroji, jako jsou například různé snímače, akční členy a jiná V/V zařízení. Čímž umožňuje rychlou komunikaci typu master-slave s vysokými nároky na rychlost a dobu odezvy.

- Profibus PA je speciální varianta sběrnice Profibus PD do výbušného prostředí. Oproti Profibusu PD má nižší přenosovou rychlost (31.25 kbit/s) a také jinou fyzickou vrstvu, kvůli použití ve výbušném prostředí a s ní přímo související část linkové vrstvy. Jako fyzické vrstvy je využito MBP popsané v normě IEC 61158-2. Vzhledem k tomu že PA i PD využívají stejný protokol je možné je pomocí spojky spojit. V tomto případě je mnohem rychlejší PD využíván jako páteřní síť pro několik segmentů PA.

Profibus sice vychází z modelu ISO/OSI, ale vzhledem k efektivnosti a rychlosti komunikace nedefinuje všech sedm jeho vrstev, nýbrž jen tři - vrstvu první fyzickou, dále vrstvu druhou linkovou a nakonec vrstvu sedmou aplikační.



Obr. 4.1 ISO/OSI model sběrnice ProfiBus

Dále se budu zabývat už jen popisem Profibusu FMS a PD, protože Profibus PA nemá jako fyzickou vrstvu RS485, ale MBP. Tento standard se však nevyskytuje ani v jednom zařízení. Z tohoto důvodu nemá cenu se o něm dále zmiňovat.

4.2.2.1 Fyzická vrstva

Profibus FMS a PD, jak ukazuje Obr. 4.1, mají stejnou fyzickou vrstvu. Nejčastěji se u těchto variant Profibusu setkáme s rozdílovou napěťovou sběrnicí podle standardu RS-485. Tato varianta je většinou vybrána pro svoji jednoduchost, nízkou cenu a vysokou přenosovou rychlost. Pro propojení se standardně používá stíněný kabel s měděným krouceným párem, jenž má následujícími parametry:

- impedance: 135 Q až 165 Q
- kapacita: < 30 pF/m
- odpor smyčky: 110 Q/km
- minimální průřez vodiče: > 0,34 mm2.

Přenosová rychlost na této sběrnici dosahuje rychlosti 9.6 kbit/s až 12 Mbit/s a je nepřímo závislá na délce vedení. Maximálního rozsahu rychlostí může dosahovat jen varianta Profibus PD. Profibus FMS je omezen na maximální rychlost do 500 kbit/s. Topologie sítě je sběrnicového typu. Maximální počet stanic v síti Profibus připojených na jeden segment sběrnice RS485, je 32. Pokud chceme tento počet navýšit je potřeba pomocí opakovače připojit další segment sběrnice. Navýšit tak můžeme počet stanic až na 126. Více už není možno kvůli sedmibitové adrese stanic, která má rozsah 0 až 125. Pomocí opakovačů můžeme taky prodloužit délku sběrnice při zachováni stejné rychlosti. Jediným kritériem pro spojování segmentů pomocí opakovačů je čas potřebný pro přenos signálu mezi nejvzdálenějšími stanicemi. Proto maximální počet opakovačů je stanoven na devět.

4.2.2.2 Linková vrstva

Linková vrstva v Profibusu se nazývá Fieldbus Data Link, zkráceně FDL. Tato vrstva se u jednotlivých variant Profibusu příliš neliší. Jako metoda řízení přístupu na sběrnici je využíváno hybridní spojení metod master-slave a token passing.

4.2.2.2.1 Řízení komunikace na sběrnici

Všechny stanice vyskytující se na sběrnici jsou rozděleny na stanice typu master, většinou se jedna o PLC, nebo průmyslové PC a na stanice typu slave, jako jsou různá čidla, pohony atd. Řízení přístupu na sběrnici probíhá následujícím způsobem - stanice typu master si předávají pověření pro přístup na sběrnici, takzvaný "token". Komunikace se stanicí typu master držící toto pověření potom probíhá metodou master-slave. Předávaní pověření mezi stanicemi typu master probíhá vzestupně podle jejich adresy, proto každá stanice typu master si sestavuje a udržuje seznam LAS (List of Active Stations). Jedná se o seznam všech adres stanic na sběrnici, kde je u každé adresy uvedeno, jestli se jedná o stanici typu master nebo slave, a z kterého zjistí, od které stanice bude přijímat pověření a které stanici má pověření následně zaslat. Aby bylo zajištěno, že žádná stanice master nebude čekat na pověření příliš dlouho, je v každé sběrnici Profibus definovaná žádaná doba oběhu pověření značená T_{TR} (Target Rotation Time). Zároveň každá stanice master měří skutečnou dobu oběhu pověření značenou T_{RR} (Real Rotation Time). Následně po přijetí pověření stanicí, dojde k porovnání T_{TR} a T_{RR}. Pokud je T_{RR} menší než T_{TR}, může stanice realizovat své komunikační požadavky. Jakmile už nemá žádné další komunikační požadavky, nebo doba dosáhla T_{TR}, předá stanice pověření svému následovníkovi. Pokud je T_{RR} větší než T_{TR} může stanice odvysílat pouze jednu komunikační zprávu a následně musí hned předat pověření následující stanici. Po každém předání požadavku následující stanici, nuluje stanice svůj časovač pro měření T_{RR}.

Rozdíly u jednotlivých verzí Profibusu na linkové vrstvě jsou následující. U Profibusu FMS může stanice držící oprávnění komunikovat jak se slave stanicemi, tak i se stanicemi master, naproti tomu u verze Profibus PD probíhá komunikace většinou jen mezi master stanici a slave stanicemi. Komunikace mezi stanicemi master je u varianty Profibus PD omezena jen na předávání pověření. Jedinou výjimkou je komunikace se stanicemi typu PDM2, jelikož u Profibusu PD existuji dva typy stanic master:

- DPM1 (DP Master Class 1) je stanice typu master, která provádí cyklickou komunikaci s podřízenými stanicemi typu slave.

- DPM2 (DP Master Class 2) je řídicí stanice realizující konfigurační, diagnostické a monitorovací funkce. Je většinou používána při uvádění sítě do provozu, případně při hledání chyb, nebo při změně konfigurace sítě.

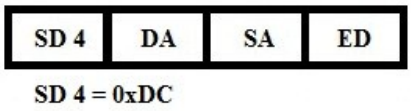
4.2.2.2.2 Formát rámce

Rámce na sběrnici Profibus jsou sestavovány z jedenáctibitových znaků, které se skládají z jednoho start bitu, osmi datových bitů, paritního bitu a jednoho stop bitu. Důležitým požadavkem na sestavování rámců na Profibusu je, aby mezi následující znaky rámce nebyly žádné časové mezery. Zabezpečení přenášených dat je na úrovni znaku řešeno pomocí sudé příčné parity a ještě na úrovni rámce podélnou paritou. Profibus definuje čtyři základní rámce:

Obr. 4.2 Rámec bez dat

Obr. 4.3 Rámec s proměnnou délkou dat

Obr. 4.4 Rámec s pevnou délkou dat 8 byte



Obr. 4.5 Rámec s pověřením k vysílání, token

Zkratka	Popis
DA	Destination Address
DSAP	Destination Service Access Point
ED	End Delimiter
FC	Function Code
FCS	Frame Checking Sequence
LE	Length of protocol data unit
LEr	Repetition of protocol data unit
SA	Source Address
SD	Start Delimiter
SSAP	Source Service Access Point

Tabulka 4.1 Popis zkratek rámce Profibus

4.2.2.3 Aplikační vrstva

Aplikační vrstva je u Profibusu FMS rozdělena na dvě podvrstvy. Nižší podvrstva se nazývá LLI (Lower Layer Interface) a tvoří rozhraní mezi linkovou vrstvou a aplikační. Zároveň tato vrstva realizuje část funkcí chybějící třetí až šesté vrstvy modelu ISO/OSI. K důležitým funkcím, které LLI provádí, patří vytváření logických komunikačních kanálů. Tyto logické kanály jsou pro každé zařízení definovány v tabulce komunikačních vztahů CRL (Communication Reference List). Tabulka bývá u jednodušších zařízení definována pevně výrobcem, u komplexnějších zařízení může být konfigurována uživatelem.

Vyšší aplikační podvrstva Profibusu FMS se nazývá FMS a jedná se o vlastní aplikační vrstvu, která poskytuje služby uživatelským programům. Popisuje komunikační objekty a služby definované nad těmito objekty. Služby podvrstvy FMS neadresují přímo zařízení, se kterým pracují, ale jeho komunikační objekt. Tyto komunikační objekty jsou sdruženy do virtuálního zařízeni VFD (Virtual Field Device) a ke každému objektu z daného VFD je přiřazen popis, který je uložený v lokálním slovníku objektu OD (Object Dictionary). VFD je abstraktní model popisující data, datové struktury a chování určitého zařízení z pohledu jeho komunikačních partnerů a představuje tak část zařízení, která je pomocí komunikačního vztahu dosažitelná. Služby poskytované FMS jsou rozdělené celkem do sedmi kategorií:

- **Context Management** – služby pro vytváření a rušení logických kanálů.
- **Variable Access** – služby pro čtení a zápis proměnných, polí, záznamů a seznamů proměnných.
- **Domain Management** – služba pro zavádění a čtení paměťových oblastí.
- **Program Invocation Management** - služby sloužící k spouštění, zastavování a mazání programů.
- **Event Management** – služby sloužící pro přenos výstražných hlášení a mohou být poslány i více účastníkům zároveň jako zprávy broadcast či multicast.
- **VFD Support** – služby umožňující přístup k virtuálním zařízením, k jejich identifikaci a zjišťování stavu.
- **OD Management** – služby pro management adresáře objektů.

U Profibusu PD představují aplikační vrstvu doplňkové funkce DP-V0, DP-V1 a DP-V2. Tyto funkce byly do standardu Profibusu postupně přidávány, takže nemusí být podporovány staršími zařízeními.

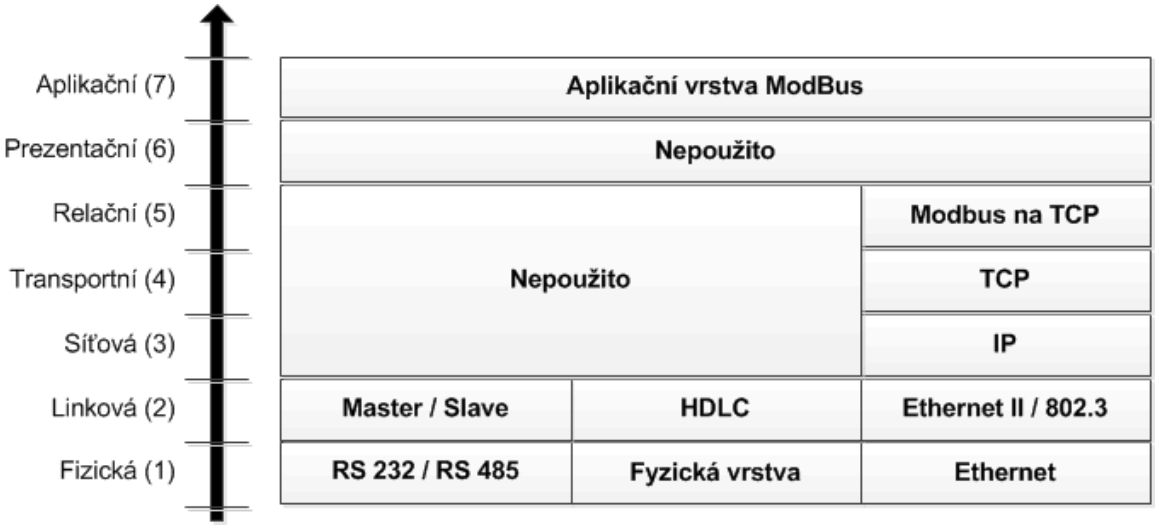
Verze DP-V0 poskytuje základní funkce komunikačního protokolu. To zahrnuje zejména cyklickou komunikaci se zařízeními, moduly a specifikaci kanálů a jejich diagnostiku pro rychlou lokalizaci poruchy.

Verze DP-V1 doplňuje DP-V0 o funkce pro acyklickou komunikaci, jako jsou funkce pro parametrizaci, provoz, monitoring a alarmy. DP-V1 umožňuje online přístup k uzlům sběrnice pomocí technických nástrojů určených pro tyto účely.

Verze DP-V2 obsahuje další funkce, které rozšiřují DP-V1. Převážně se jedná o funkce, které jsou potřeba pro řízení pohonů. Patří mezi ně funkce pro komunikaci mezi slave zařízeními, jako je synchronizace cyklu a času.

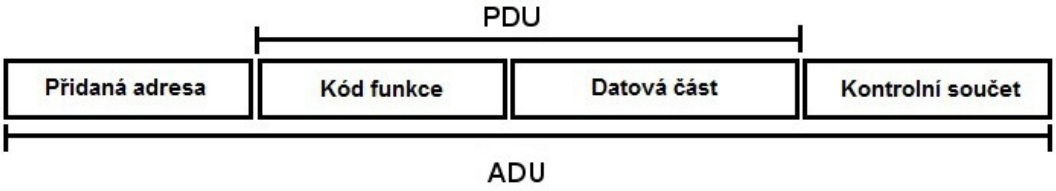
4.2.3 ModBus

ModBus[8] je komunikační protokol sedmé aplikační vrstvy modelu ISO/OSI, který poskytuje klient/server komunikaci mezi zařízeními na různých typech sběrnic a sítí. V oblasti sériových sítí je v podstatě standardem od roku 1979, kdy byl vytvořen firmou MODICON. V současné době ModBus podporuje celou řadu komunikačních médií, jako jsou například sériové linky typu RS232, RS422 a RS485, optické a radiové sítě, nebo Ethernetové sítě s využitím protokolu TCP/IP. Komunikace mezi zařízeními probíhá formou požadavku a odpovědi a druh požadované funkce je specifikován v kódu funkce, který je součásti požadavku.



Obr. 4.6 ISO/OSI model protokolu ModBus

Protokol ModBus definuje strukturu zprávy označovanou jako PDU nezávisle na typu komunikační vrstvy. V závislosti na typu sítě, nebo sběrnice, na které je protokol používán, se PDU část rozšiřuje o další části, tato zpráva se nazývá ADU.



Obr 4.7 Obecný rámec protokolu ModBus

ADU je sestavována na straně klienta, který iniciuje ModBus transakci. Kód funkce určuje jaký druh operace má server provést. Platné kódy funkcí jsou v rozsahu 1 až 255, přičemž rozsah 128 až 255 je vyhrazen pro chybové odpovědi. Některé kódy funkcí obsahují i kód podfunkce upřesňující blíže požadovanou operaci. Datová část zprávy, poslaná klientem na server, obsahuje dodatečné informace potřebné k vykonání požadované operace určené kódem funkce. Datová část může tedy obsahovat adresu registru a počet skutečných datových bajtů v této oblasti, které má server přečíst, nebo adresu registru a hodnotu nutnou k zapsání do daného registru. Některé funkce doplňující informace nepotřebují, proto u těchto funkcí může datová část ve zprávě úplně chybět. Nedojde-li při zpracování požadavku k chybě, odpoví server zprávou, která má stejný kód funkce, jako měla zpráva s požadavkem. Datová část potom obsahuje požadovaná data, pokud měla být nějaká předána. Jestliže dojde při zpracování požadavku k chybě, odešle server klientovi zprávu, která v poli kódu funkce obsahuje kód požadované funkce zvětšený o 128. Datová část potom obsahuje chybový kód upřesňující důvod neúspěchu.

Velikost ModBus PDU je omezena velikostí zděděnou od prvního provedení ModBusu na sériové linceRS485, kde byla maximální velikost ADU 256 bytů. Tomu odpovídá maximální velikost PDU 253 bytů. Velikost ADU na TCP/IP je tedy velká 260 bytů.

ModBus zakládá svůj datový model na sadě čtyř základních tabulek s charakteristickým významem. Tyto tabulky jsou popsány v tabulce 4.2

Tabulka	Typ Objektu	Přístup	Popis
Diskrétní vstupy	Jeden bit	Pouze čtení	Data poskytovaná I/O systémem
Cívky	Jeden bit	Čtení - Zápis	Data modifikovatelná aplikačním programem
Vstupní registry	16-bitové slovo	Pouze čtení	Data poskytovaná I/O systémem
Uchovávací registry	16-bitové slovo	Čtení - Zápis	Data modifikovatelná aplikačním programem

Tabulka 4.2 ModBus datový model

Mapování tabulek do adresního prostoru určuje výrobce samotného zařízení. Každá z tabulek může mít vlastní adresní prostor, nebo se mohou částečně či úplně překrývat. Každá ze základních tabulek může mít podle protokolu až 65 536 položek. Čtení a zápis těchto položek je možný provádět jednotlivě, nebo je možné provádět tyto operace nad celými skupinami. Velikost této skupiny je omezena maximální velikostí datové části zprávy. V ModBus zprávách v části PDU jsou datové položky adresovány od 0 do 65 535. To také

definuje jasně ModBus datový model složený ze 4 bloků, které zahrnují několik prvků, číslovaných od 1 do n.

ModBus definuje celkem tři základní skupiny kódů funkci, kódy funkcí veřejných, kódy funkcí uživatelských a kódy funkcí rezervovaných. Rezervované kódy funkcí jsou v současnosti používány některými firmami a nejsou dostupné pro veřejné použití. Funkce kódu veřejné a uživatelské mají rozsah kódů 0 až 127 rozděleny následovně:

- Veřejné kódy funkcí 0 - 64 a 73 - 99 a 111 - 127
- Uživatelsky definované kódy 65 - 72 a 100 - 110

4.2.3.1 Vybrané kódy funkcí

Vybral jsem funkce 01, 02, 03, 04, 05, 06, 08 (podfunkce 00), které podporuje snímač váhy

- Funkce 01 Čti cívky a 02 Čti diskrétní vstupy. Tyto funkce se liší jen typem tabulky, které čtou. Obě umožňují čtení stavu 1 až 2 000 položek. V požadavku je určena adresa první položky a počet položek, které se mají přečíst. V odpovědi je v jednom bytu přenášen stav celkem 8 položek. Nejnižší bit prvního bytu je stav první adresované položky.
- Funkce 03 Čti uchovávací registry a 04 Čti vstupní registry. Tyto funkce se liší jen typem tabulky, které čtou. Obě funkce umožňují číst souvislý blok registru ve vzdáleném zařízení. V požadavku je určena adresa prvního registru a počet registrů, které se mají přečíst. V odpovědi každému čtenému registru odpovídá dvojice bytů. Pro každý registr obsahuje první byte vyšší bity a druhý byte nižší bity.
- Funkce 05 Zapiš jednu cívku. Tato funkce slouží k nastavení jednoho výstupu ve vzdáleném zařízení na ON nebo OFF. V požadavku je určena adresa výstupu, který se má nastavit a hodnota na kterou se má výstup nastavit. Hodnoty jsou povoleny jenom dvě, 0 x 0000 pro OFF a 0 x 00FF pro ON, ostatní hodnoty jsou brány jako ilegální. V odpovědi je kopie požadavku.
- Funkce 06 Zapiš jeden registr. Tato funkce slouží k zápisu jednoho uchovávacího registru ve vzdáleném zařízení. V požadavku je určena adresa registru, který se má zapsat a hodnota, která se má zapsat. V odpovědi je kopie požadavku.
- Funkce 08 Diagnostika. Tato funkce slouží k testování komunikace mezi klientem a serverem, nebo pro kontrolu různých interních chybových stavů serveru. V požadavku je určen dvoubajtový kód podfunkce definující požadovaný test. V odpovědi je kopie požadavku, případně další data pokud jsou výsledkem testu. Podfunkce 00, příchozí požadavek je odeslán jako odpověď, slouží jako test komunikace.

4.2.4 Konzultace řešení

Po prostudování vybraných protokolů a sběrnic jsem další postup konzultoval s mým konzultantem. Jako upřednostňované řešení bylo určeno získávaní požadovaných hodnot ze vzdáleného displeje, který je umístěn přímo ve výrobní hale. Toto řešení bylo nakonec po konzultaci zamítnuto. Vzhledem k tomu, že hlavním požadavkem byla komunikace pomocí firemní sítě a pro připojení tohoto zařízení do firemní sítě, bylo nutné zakoupit Ethernetovou bránu pro Profibus, nebo ji vyrobit. Byla tedy zvolena druhá možnost a to připojit do firemní sítě snímač váhy, který je přímo vybaven ethernetovým konektorem. Toto řešení bylo zvoleno také díky menší finanční náročnosti. Pro komunikaci mezi programem a snímačem hmotnosti,

v tomto případě, slouží protokol ModBus TCP. Dále bylo ještě navrhnuto nouzové řešení v podobě využití sériového portu, který by byl pomocí převodníku připojen do firemní sítě. Tento port se měl nastavit jako tiskový pro cyklické zasílání hodnot o váze a program by měl za úkol tato data zachytit a zpracovat.

4.3 Implementace a testováni

Jako první krok při vytváření nového modulu jsem měl ověřit samotnou funkčnost vybraného protokolu ModBus. Pro tento účel jsem napsal konzolovou aplikaci, která implementuje pouze samotné připojení k zařízení, a komunikaci sním. Pro tento účel jsem využil knihovny jamod (http://modbus.org/tech.php). Toto řešení jsem zvolil pro lepší ověření funkčnosti daného protokolu na ethernetovém portu snímače váhy. Tato knihovna poskytuje velké množství tříd pro vytvoření, jak zařízení typu master, tak slave. Z těchto tříd jsem využil pouze třídy pro zařízení typu master. Pro implementaci *ReadInputRegistersRequest*. tohoto typu zařízení poskytuje knihovna několik tříd pro vytvoření spojení se zařízením master. Každá třída implementuje jiný typ spojení daný typem protokolu ModBus, například ModBus RTU pro sériovou linku, nebo ModBus TCP pro sítě ethernet. V testovací aplikaci jsem využil třídy *TCP MasterConnection* a *UDP MasterConnection*, sloužící k vytvoření spojení pomocí protokolu TCP, nebo UDP. Jako parametr se těmto třídám předává IP adresa slave zařízení, pomocí metody setPort(int port) se musí ještě určit port, standartní port pro ModBus je 502. Dále knihovna poskytuje třídy pro tvorbu transakcí, protože tyto třídy vytvářejí samotný ADU rámec, je každému typu protokolu ModBus přiřazena jedna třída. Pro ModBus TCP a UDP jsou to třídy *ModbusTCPTransaction* a *ModbusUDPTransaction*. U těchto tříd je jako parametr předáno vytvořené spojení a pomocí metody setRequest(ModbusRequesreq) je k transakci přiřazen požadovaný požadavek. K dalším důležitým metodám této třídy patří metoda execute() vykonávající transakci a metoda getResponse() vracející odpověď od slave. Poslední třídou z knihovny jamod, použitou v testovací aplikaci, byla třída *ReadInputRegistersRequest*. Tato třída vytváří PDU rámec kódu funkce 4, pro čtení vstupních registru. Jako parametr je předána adresa registru a počet registrů ke čtení. Kromě třídy *ReadInputRegistersRequest* obsahuje knihovna ještě další třídy realizující kódy funkce: 1, 2, 3, 5, 6, 15 a 16.

Testovací aplikaci pro ověření funkčností ModBus protokolu tvořila jedna třída s třemi metodami Connection(), Disconnection() a Test (). Metody Connection() a Disconnection() slouží k vytvoření a zrušení spojení se slave zařízením, metoda Test () provádí vytvoření a zaslání požadavku slave zařízení a přečtení odpovědi, kterou následně vypíše do konzole. Testovací aplikace byla vytvořená jen pro účel ověření funkčnosti komunikačního protokolu ModBus a nepočítalo se s dalším použitím. Proto veškeré potřebné údaje pro vytvoření požadavku a spojení jsou zapsané přímo do proměnných ve třídě, ale pokud by byla potřeba, dal by se tento program rozšířit o vstup proměnných z konzole. Z toho důvodu jsem také neprováděl na aplikaci žádné unit testy.

Zároveň s testovací aplikací pro ModBus jsem měl vytvořit ještě jednu jednoduchou aplikaci, pro čtení příchozích dat ze sériové linky, aby bylo možné zjistit formát zasílaných dat ze snímače hmotnosti na tuto linku. Pro tuto aplikaci jsem potřeboval knihovnu s třídami podporujícími komunikaci po sériové lince, protože Java verze 1.7 tyto třídy nenabízí. Zvolil jsem tedy pro tento účel velmi rozšířenou knihovnu RXTXcomm (http://rxtx.qbang.org). Aplikaci pro čtení dat ze sériové linky tvoří jedna třída s dvěmi metodami. Metoda Open()

vytváří spojení s daným sériovým portem, nastavuje jeho parametry pro komunikaci a také na tomto spojení registruje nového posluchače události a vytváří vstupní proud. Metoda serialEvent (SerialPortEventspe) slouží k zpracování událostí na sériové lince, při vyvoláni události DATA_AVAILABLE dojde k čtení dat ze vstupního proudu a následného vypsání těchto dat na konzoly. Stejně jako u testovací aplikace pro ModBus jsou všechny potřebné údaje pro nastavení spojení se sériovým portem a nastavení jeho parametru zapsané přímo do proměnných ve třídě.

Než jsem mohl přistoupit k samotnému testování, musel jsem nejdříve nakonfigurovat síťové spojení mezi notebookem a snímačem váhy. Toto spojení, jak už bylo řečeno, je také nutné k samotné konfiguraci tohoto zařízení pomocí programu VNC. Zde se ale objevil problém. Zařízení je standardně nastaveno tak, aby získávalo adresu z DHCP serveru a pokud není DHCP server k dispozici, přepne se zařízení automaticky na nastavenou statickou adresu. Problém nastal v tom, že statická IP adresa by měla být zapsaná na štítku na krytu přístroje, bohužel tento štítek byl nevyplněn. Tuto situaci jsem vyřešil instalací programu, který na notebooku vytvořil DHCP server. Tento program jsem před použitím na snímači váhy úspěšně otestoval, při vytvoření spojení mezi dvěma počítači. Po úspěšném nastavení sítě mezi notebookem a snímačem váhy jsem dále pokračoval v jeho konfiguraci pomocí programu UltraVNC (http://www.uvnc.com). Zaprvé jsem z nastavení síťového připojení zjistil statickou IP adresu zařízení, kterou jsem si zapsal pro další použití. Zadruhé jsem v nastavení portu aktivoval na portu RS232 funkci printer a nastavil přenosové parametry, jako je rychlost přenosu, protokol, paritu, stop bit a velikost rámce. Dále jsem ještě musel nastavit parametry printer funkce. Parametry jsem nastavil tak, aby funkce prováděla cyklické zasílání vybrané položky. Jako položku jsem vybral hmotnost granulátu a dobu cyklu jsem nastavil na 10 sekund.

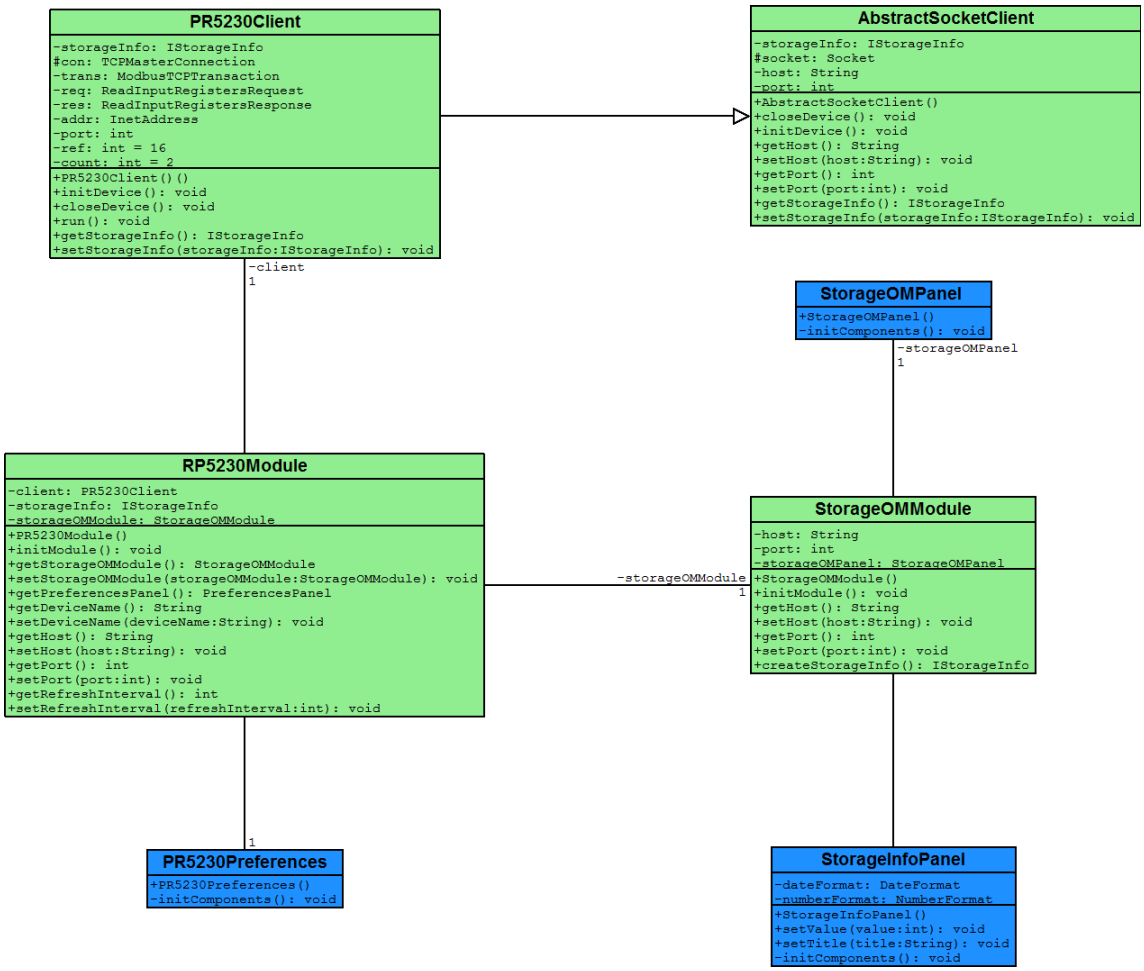
Po nastavení potřebných parametrů jsem přistoupil k testu pomocí připravených aplikací. Nejprve jsem vyzkoušel funkčnost protokolu ModBus UDP. Tento protokol sice na diagramu podporovaných protokolu chybí, ale v části dokumentace věnované protokolu ModBus je o něm zmínka[4]. Chtěl jsem tedy ověřit, jestli tento protokol je, nebo není podporovaný, protože ModBus UDP je vhodnější varianta pro rychlou cyklickou výměnu dat, tedy přesně to co měla aplikace dělat. Test ale ukázal, že protokol ModBus UDP opravdu nepatří mezi podporované protokoly snímačem váhy. Jako další test jsem provedl ověření podpory protokolu ModBus TCP. Na rozdíl od testu varianty UDP tento test proběhl úspěšně. Na konzoli byla aplikací vypsaná příchozí data v podobě hmotnosti v kilogramech. Správnost zobrazené hodnoty aplikací, jsem hned ověřil pomocí porovnání údaje zobrazovaného na display snímače váhy. Oba dva údaje se shodovaly až na jednotky, které se ale neustále měnily. Test jsem provedl ještě několikrát, abych měl větší jistotu ve správnou funkčnost aplikace. Otestoval jsem i variantu aplikace, která četla data ze sériové linky. Tato aplikace také fungovala správně. Přišlé bity bylo bez problému možné převést na čitelný string a zobrazit na konzoli. Formát zobrazovaného stringu byl číselná hodnota s údajem o jednotce hmotnosti. Bylo by teda možné využít čtení těchto cyklicky zasílaných dat a vyparsovat z nich potřebná data. Pro modul jsem ale zvolil variantu získávání dat pomocí protokolu ModBus TCP z důvodu efektnějšího získávání dat a také nebylo potřeba vynakládat finanční prostředky na pořízení převodníku.

Po otestování a rozhodnutí o konečném způsobu komunikace jsem začal tvořit zásuvný modul pro framework. Tento modul má spolupracovat s již vytvořeným modulem StorageOMModule, sloužící k výpisu získaných dat do grafického prostředí programu. Modul má také využívat program běžící jako služba na serveru, tento program slouží k ukládaní získaných dat do databáze ERP systému. Jak už bylo řečeno, pro tvorbu modulu jsem využil několik tříd z frameworku. Zásuvný modul tvořily celkem tři třídy. Hlavní třída pojmenovaná *PR5230Module* dědí z abstraktní třídy *AbstractModule*, která je součástí frameworku. Zděděním této abstraktní třídy určujeme, že tato třída představuje modul. Další třídou modulu je třída *PR5230Preferences* dědící z třídy *PreferencesPanel* typu JPanel. Poslední třídou modulu je třída *PR5230Client* dědící z abstraktní třídy *AbstractSocketClient*, která je potomkem abstraktní třídy *AbstractDevice* z frameworku.

Hlavní třída *PR5230Module* slouží k vytvoření instancí třídy *PR5230Preferences* a *PR5230Client* a k vytvoření nového info panelu, který bude zobrazovat získaná data. Třída obsahuje několik metod, mezi nejdůležitější patří překrytá metoda initModule(). Tato metoda vykonává inicializování některých proměnných třídy *PR5230Client* a voláním metody startInit() zahájí inicializaci samotného klienta. Druhou důležitou metodou je překrytá metoda getPreferencesPanel(), která vytvoří nový panel možností. Dále třída obsahuje už jen konstruktor a getry a setry.

Třída *PR5230Preferences* představuje grafické rozhraní pro nastavení vstupních proměnných. Tím, že dědí ze třídy *PreferencesPanel,* je určeno, že se grafické rozhraní vytvořené v této třídě zobrazí jako záložka v okně možností. Grafické rozhraní obsahuje textová pole pro zadání IP adresy zařízení slave a názvu zařízení. Součástí rozhraní jsou ještě dva spinery pro zadání portu a obnovovacího intervalu. Důležité bylo, při vytváření rozhraní dodržet parametr propertyName, aby se shodoval s názvem proměnné v abstraktní třídě *AbstractSocketClient*. Nedodržením těchto názvu by nedošlo k načtení zadaných hodnot pomocí frameworku do proměnných modulů.

Třída *PR5230Client* obsahuje samotnou komunikační část modulu. Pro mé účely jsem v této třídě musel překrýt metody initDevice() a closeDevice() z rodičovské třídy. Tyto metody jsou volány pro vytvoření a ukončení spojení. Původní moduly využívaly pro vytvoření spojení socketu, proto jsem musel tyto metody překrýt novými, které využívají pro vytvoření spojení tříd z knihovny jamod. Kromě těchto metod obsahuje třída ještě konstruktor jednu metodu typu get a set a metodu run () spouštěnou jako vlákno. V metodě run() je obsažena samotná komunikační část, která byla popsaná u testovací aplikace. Jediný rozdíl je ve zpracování přijatých dat. Data nejsou vypisována na konsoly, ale jsou předána pomocí interface buď grafickému rozhraní, nebo aplikaci pro jejich ukládaní do databáze.



Obr. 4.8 Class Diagram Modulu

Testování funkčnosti hotového modulu jsem nejprve provedl pomocí programu ModBus PLC Simulator. Program slouží k simulaci slave zařízení a podporuje jak variantu ModBus RTU tak variantu TCP. Tento program umožňuje zobrazení terminálu, na kterém je vypsaná veškerá komunikace, kterou program provedl. Jsou zde vypisovány veškeré přijaté a odeslané rámce. Pomocí tohoto programu jsem si ověřil správnost zasílaných rámců i funkčnost celého modulu. Takto otestovaný modul jsem ještě otestoval přímo na zařízení snímače váhy, kde test proběhl úspěšně.

Posledním mým úkolem bylo, po připojení snímačů hmotnosti do podnikové sítě ethernet, nakonfigurovat jejich síťové parametry. Pomocí programu UltraVNC jsem tedy nastavil mně zadané IP adresy masky podsítě a výchozí brány. Po nakonfigurování parametru jsem provedl ještě jeden test funkčnosti komunikace. Tentokrát po podnikové síti jsem zároveň otestoval komunikaci i s druhým, novým silem. Oba testy proběhly v pořádku, tak jsem mohl předat modul k užívání.

5 Uplatněné znalosti a dovednosti

V průběhu praxe jsem využil znalostí získané během studia předmětů programovací jazyky I, Úvod do programování a Programovací techniky. Díky studiu těchto předmětu jsem byl schopen snáze nastudovat samotné prostředí a funkčnost frameworku. Zároveň jsem nabyté znalosti uplatnil při tvorbě programu. U tvorby uživatelského rozhraní pro panel možností jsem mohl zúročit znalosti získané během studia předmětu uživatelská rozhraní. Tyto znalosti mi umožnily bezproblémovou tvorbu uživatelského rozhraní. Nebýt těchto znalostí, musel bych během praxe nastudovat práci s objekty knihovny Java Swing.

Při vykonávání praxe jsem ještě využil znalosti získané z předmětu Architektura Počítačů. Jednalo se o znalosti parametru a zapojení sériové linky, které jsem využil při tvorbě testovací aplikace i při samotném testování. Dále pak některé znalosti získané při studiu předmětu Počítačové sítě, které mi hlavně pomohly při nastavování síťového spojení.

Navíc jsem využil znalost různých typů komunikace, jako je například komunikace typu Master Slave nebo Token Ring. Tuto znalost jsem hlavně využil při studování komunikačních protokolů a díky ní jsem snáze mohl pochopit samotné protokoly. Užitečná byla i znalost referenčního modelu OSI, který se hojně vyskytuje v dokumentacích pro komunikační protokoly, případně i u popisu sběrnic.

6 Scházející znalosti a dovednosti

Mezi znalosti, které mi během vykonávání praxe hlavně scházely, patřila znalost automatizační komunikačních protokolů a sběrnic. I když z předchozího studia jsem měl základní znalosti automatizační techniky, tak bohužel znalosti komunikačních protokolu a sběrnic mezi nimi chyběly. Proto jsem musel získat informace o těchto protokolech a sběrnicích a ze získaných informací si rozšířit znalosti, abych mohl navrhnout vhodné řešení pro splnění zadaného úkolu. Jednalo se o komunikační protokol ModBus a průmyslovou sběrnici Profibus využívanou u řídících a měřících přístrojů sila.

Další znalosti, které jsem během praxe získal, už jen rozšiřovaly mé dosavadní znalosti získané během studia. Jednalo se převážně o seznámení se s novými technologiemi využívanými v praxi, jako je technologie Java Web Start, či různé rozšiřující knihovny pro jazyk Java. Nová pro mě byla i zkušenost vývoje aplikace pomocí frameworku, který mi díky využití jeho tříd a metod velmi usnadnil a zrychlil vývoj samotné aplikace.

Úplně neznámou technologií, s kterou jsem se během praxe setkal a využíval ji, byla technologie VNC pro dálkové sdílení uživatelské plochy pomocí počítačových sítí. Tato technologie mi přišla pro průmyslové využití zajímavá z toho důvodu, že je tak možné centrálně nastavovat parametry různých zařízení rozmístěných na různých místech. Toto řešení tak šetří čas a zefektivní práci.

Kromě znalostí získaných přímo pro potřeby splnění úkolu, jsem si rozšířil znalosti i o informace týkající se služeb a s nimi souvisejícími technologiemi, kterými se firma zabývá

a které přímo nesouvisely s plněním mého úkolu. Jednalo se například o využití virtuálních serverů pro zefektivnění využití výpočetní kapacity serveru nebo o informace k užívaným systémům pro podporu uživatele.

7 Dosažené výsledky a zhodnocení

Výsledkem mé práce při vykonávání praxe je nový programový modul pro dálkový odečet hodnot ze snímačů váhy nových sil. Tento modul byl v poslední den mé praxe zařazen k stávajícím modulům a společně s nimi plní úspěšně funkci průběžného sběru dat ze všech sil ve firmě. Proto si myslím, že moje praxe přinesla firmě přínos a samotný program svou funkcí usnadňuje a zefektivňuje práci některým zaměstnancům.

Zároveň měla praxe velký přínos i pro mě samotného. Mohl jsem tak využít znalosti nabyté v průběhu studia přímo při řešení praktického úkolu. Zároveň jsem prakticky využil věci, které jsem znal jen z teoretického hlediska. Udělal jsem si taky obraz o tom, jak se v praxi přistupuje k řešení úkolu a jak se při jeho řešení postupuje. Dále jsem si musel osvojit prezentování mých návrhů a rozhodnutí při konzultaci s vedoucím mé praxe. Musel jsem ze získaných informací stanovit klady a zápory a navrhnout nejvhodnější řešení problému. Dále pro mě byla zajímavá práce v kolektivu lidí, s kterými jsem mohl konzultovat případné problémy a přijímat rady, jak by mohl jít konkrétní problém vyřešit. Tyto všechny nově nabyté znalosti a zkušenosti mi mohou pomoci při vykonávání budoucího zaměstnání, případně zmínka o vykonané praxi během studia mi může pomoci u přijímacího řízení na nové pracovní místo.

8 Závěr

Výsledkem bakalářské práce je popis průběhu mé odborné praxe. Při popisu jsem se soustředil na popsání mnou vykonané práce, znalostí a dovedností, které jsem během praxe využil, nebo naopak musel získat. Zároveň jsem do popisu praxe zařadil popis některých technologii, které pro mě byly nové a které jsem během praxe nastudoval a využil. Při popisu programu jsem se zaměřil hlavně na popsání funkce jednotlivých tříd a jejich metod, proto v práci nejsou obsažené žádné ukázky zdrojového kódu.

Při realizaci zadaného úkolu jsem narazil na problém neznalosti průmyslových komunikačních protokolů a sběrnic. Tyto chybějící znalosti jsem si během praxe doplnil a zároveň hned zúročil při plnění úkolu.

Výsledný program by se dal dále rozšířit o další modul pro grafické znázornění zaplnění sil, nebo o modul pro vykreslování grafů spotřeby granulové hmoty ze sil v různých časových období.

Nakonec bych zhodnotil vykonanou praxi jako pro mě prospěšnou. Během praxe jsem měl možnost vidět uplatnění mých znalostí. Zároveň jsem během praxe získal i nové znalosti, které budu moci uplatnit během plnění budoucích úkolů. Také jsem se během praxe seznámil s novými lidmi z oboru a získal představu, jak vypadá práce v mezinárodní firmě.

Požitá literatura

[1] Java Web Start. PATRNÝ, Vojtěch. *Linuxzone* [online]. 2003 [cit. 2012-04-20]. Dostupné z:http:/www.linuxzone.cz/index.phtml?ids=2&idc=637

[2] Java™ Web Start Guide. *Oracle* [online]. 2011 [cit. 2012-04-20]. Dostupné z: http://docs.oracle.com/javase/6/docs/technotes/guides/javaws/developersguide/contents.html

[3] ZEPPELIN SILOS & SYSTEMS GMBH. *SCS Silo Control System*: *Operatinginstructions*. 1.0. Friedrichshafen, 2011.

[4] SARTORIUS MECHATRONICS T&H GMBH. *Transmitter in Field Housing PR 5230*: *Instrument Manual*. 2.00. Hamburg, 2011.

[5] PROFIBUS NUTZERORGANISATION E.V. *PROFIBUS System Description - Technology and Application* [online]. 4.332. 2010, 28 s. Dostupné z: http://www.profibus.com

[6] Trnka, P. *Profibus DP Master na PC*. Praha, 2004. 70s. Diplomová práce na Elektrotechnické fakultě Českého vysokého učení technického na katedře Řídicí techniky. Vedoucí diplomové práce Ing. Petr Smolík

[7] KOZIOREK, Jiří a Jindřich ČERNOHORSKÝ. *Distribuované systémy řízení*: *Sylaby k předmětu distančního studia*. Ostrava, 2002, 95 s. 41-45.

[8] MODBUS ORGANIZATION, Inc. *MODBUS Application Protocol Specification* [online]. V 1.1b. 2006, 51 s. Dostupné z: http://www.modbus-ida.org/specs.php